**Variable Arguments**

# Variable Arguments

C supports variable numbers of arguments. But there is no language provided way for finding out total number of arguments passed. User has to handle this in one of the following ways:

1. By passing first argument as count of arguments.
2. By passing last argument as NULL (or 0).
3. Using some printf (or scanf) like mechanism where first argument has placeholders for rest of the arguments.

* Define a function with its last parameter as ellipses and the one just before the ellipses is always an int which will represent the number of arguments.
* Create a va\_list type variable in the function definition. This type is defined in stdarg.h header file.
* Use int parameter and va\_start macro to initialize the va\_list variable to an argument list. The macro va\_start is defined in stdarg.h header file.
* Use va\_arg macro and va\_list variable to access each item in argument list.
* Use a macro va\_end to clean up the memory assigned to va\_list variable.

**#include <stdarg.h>**

#include <stdio.h>

// this function returns average of integer numbers passed. First argument is count of arguments.

double average(int arg\_count, ...) {

int i;

double avg = 0;

double sum = 0;

// va\_list is a type to hold information about variable arguments

va\_list ap;

// va\_start must be called before accessing variable argument list

va\_start(ap, arg\_count);

// Now arguments can be accessed one by one using va\_arg macro

// traverse rest of the arguments to find out sum

for(i = 0; i < arg\_count; i++) {

sum = sum + va\_arg(ap, int);

}

// calculate avg

avg = (sum / arg\_count);

//va\_end should be executed before the function returns whenever

// va\_start has been previously used in that function

va\_end(ap);

return avg;

}

int main() {

printf("avg value is %f\n", average(3,5,10,15));

printf("avg value is %f\n", average(4,2,3,4,5));

return 0;

}

Output:

avg value is 10.000000

avg value is 3.500000

# END